



11 Publication number : **0 574 245 A2**

12 **EUROPEAN PATENT APPLICATION**

21 Application number : **93304476.0**

51 Int. Cl.<sup>5</sup> : **G06F 15/62**

22 Date of filing : **09.06.93**

30 Priority : **11.06.92 US 897180**  
**11.06.92 US 897181**

43 Date of publication of application :  
**15.12.93 Bulletin 93/50**

84 Designated Contracting States :  
**DE FR GB**

71 Applicant : **International Business Machines Corporation**  
**Old Orchard Road**  
**Armonk, N.Y. 10504 (US)**

72 Inventor : **Moller, Christian Henrik Luja**  
**11612 Buttonwood Drive**  
**Austin, Texas 78759 (US)**

74 Representative : **Moss, Robert Douglas**  
**IBM United Kingdom Limited Intellectual Property Department Hursley Park**  
**Winchester Hampshire SO21 2JN (GB)**

54 **Method and apparatus for variable expansion and variable shrinkage of an image.**

57 Disclosed are methods and apparatus for variable magnification and for variable shrinking of an image. The techniques allow rapid zooms or rapid shrinks of images to be achieved by either hardware or software. The individual elements of the source and destination image data are correlated and then the remaining elements of the source data in the case of a shrinkage and of the destination data in the case of an expansion are distributed among the destination data or the source data respectively.

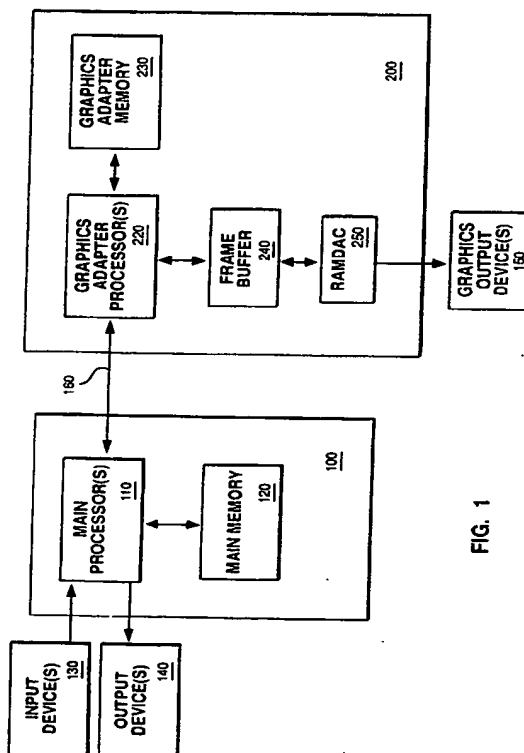


FIG. 1

EP 0 574 245 A2

Technical Field

The present invention relates to image information processing and more particularly to performing variable magnification and variable minification of an image.

Background Art

The prior art includes several methods and apparatus for expanding and for shrinking. These are described below.

U.S. Patent 4,153,896 teaches compressing or expanding an image such as a symbol, character or text, in a first direction followed by compressing or expanding the image in a second direction. The image to be compressed or expanded is divided up by a grid defining a large number of square areas in an image wherein each area is represented by a clock cycle in each direction. The compression or expansion is accomplished by utilising an averaging circuit which provides a signal indicating the value of the resulting image.

U.S. Patent 4,503,469 teaches a picture image enlarging/reducing system wherein image magnification by a desired factor is achieved by first enlarging the image by a integer factor greater than the desired factor and then multiplying the enlarged image by a correction coefficient corresponding to the ratio of the desired factor to the integer factor.

U.S. Patent 4,532,605 is directed to forming a zoom operation on a displayed image and maintaining a fixed width of each line of the zoomed image regardless of the degree of magnification of the zoomed image.

U.S. Patent 4,569,681 is directed to enlarging a binary image by inserting expansion rows and columns between previously existing rows and columns and setting the value of the expansion rows and columns equal to the value of one or both of the adjacent previously existing rows and columns.

U.S. Patent 4,631,751 is directed to enlarging a binary image by rotating the image 90 degrees, inserting expansion rows into the image, building the expansion rows based upon the values of the neighboring rows, rotating the image another 90 degrees, inserting additional blank rows, and filling those blank rows based upon the values of adjacent rows.

U.S. Patent 4,633,503 is directed to a video zoom processor by mapping each input pixel into at most four adjacent output pixels wherein each sampled input pixel is skilled, segmented and accumulated into four locations in the image memory.

U.S. Patent 4,656,664 is directed to a method for reducing a binary image by rotating the image 90 degrees, removing one or more rows of bits, modifying rows adjacent to those removed rows to preserve selected information from the discarded line, rotating the image 90 degrees, removing additional rows, and modifying the line adjacent to each line discarded to preserve selected information from the discarded line, and repeating the steps of rotating, removing and modifying a predetermined number of times to achieve a reduction by a desired factor.

U.S. Patent 4,675,830 is directed to a method for producing scalable contour data. The method stretches and compresses character contours to bring specified contour points into proper alignment with a bit map grid.

U.S. patent 4,698,778 is directed to variable magnification of half tone images such as photographs or painted pictures. This is accomplished by utilising one of a plurality of mother matrix patterns that is specified in accordance with the desired magnification.

U.S. Patent 4,754,270 is directed to generating a variable size image on a display from a matrix of horizontal and vertical pixel information stored in memory by utilising vertical and horizontal address counters which are incremented based on the desired magnification.

U.S. Patent 4,774,581 is directed to a digital video zoom system which includes circuitry for writing image samples into a memory at a standard rate and for reading samples from the memory at a reduced rate.

U.S. Patent 4,809,083 is directed to enlarging or reducing an image at a desired magnification in a facsimile, copying machine or the like. This is accomplished by continuously varying a ratio of a reading resolution to a recording density.

U.S. Patent 4,827,433 is directed to changing magnification of image data applicable to a digital copier, facsimile or the like. This is accomplished by sampling original image data at a predetermined rate based on the desired magnification.

U.S. Patent 4,833,625 is directed to an image viewing station wherein digitised image data at any size from any different modalities is stored in an original format and can be magnified or minified for display to a user. Magnification is performed by producing interpolated image data for each pixel of a magnified image based on the weighted sum of grey levels present of each pixels four nearest neighboring pixels, including pixels from other rows or columns. Minification is performed by discarding pixels to achieve the desired minification.

U.S. Patent 4,972,264 is directed to viewing and overscanned image by storing the original image as a

first data block and generating a second data block for display by systematically removing every  $n$ th data element where  $n$  is an integer less the number of data elements in the first data block.

## 5 Disclosure of the Invention

Accordingly the present invention provides a method of expanding an image stored in a first array comprising the steps of separating said first array into a plurality of rows, each row having a first plurality of data elements, each element having a value, and separately expanding each row from a first plurality of data elements in said first array to a second plurality of data elements in a second array including the steps of correlating each of said first plurality of data elements to at least one of said second plurality of data elements, and distributing the value of selected elements of said first plurality of data elements to correlated data elements of said second plurality of data elements.

The present invention also provides a method of shrinking an image stored in a first array comprising the steps of separating said first array into a plurality of rows, each row having a first plurality of data elements, each element having a value, and separately shrinking each row from a first plurality of data elements in said first array to a second plurality of data elements in a second array including the steps of correlating each of said second plurality of data elements to at least one of said first plurality of data elements, and calculating the value of each of said second plurality of data elements from the values of correlated data elements.

In addition, the present invention also provides an apparatus for expanding an image stored in a first array comprising means for separating said first array into a plurality of rows, each row having a first plurality of data elements, each element having a value and means for separately expanding each row from a first plurality of data elements in said first array to a second plurality of data elements in a second array including means for correlating each of said first plurality of data elements to at least one of said second plurality of data elements, and means for distributing the value of selected elements of said first plurality of data elements to correlated data elements of said second plurality of data elements.

Also provided is an apparatus for shrinking an image stored in a first array comprising means for separating said first array into a plurality of rows, each row having a first plurality of data elements, each element having a value and means for separately shrinking each row from a first plurality of data elements in said first array to a second plurality of data elements in a second array including means for correlating each of said second plurality of data elements to at least one of said first plurality of data elements, and means for calculating the value of each of said second plurality of data elements from the values of correlated data elements.

## Brief Description of the Drawings

The invention will now be described, by way of example only, with reference to the accompanying drawings, in which:

Fig. 1 is a block diagram of a typical digital computer and graphics adapter utilised by a preferred embodiment of the invention;

Fig. 2 is a block diagram illustrating the layers of code typically utilised by the digital computer and graphics adapter of Fig. 1 to perform graphics functions;

Fig. 3 is an illustration of an image having three rows and five columns being enlarged to an image having four rows and eight columns according to one embodiment of the invention;

Fig. 4 illustrates an expansion of a row of the images of Fig. 3;

Fig. 5 is an illustration of an image having three rows and five columns being enlarged to an image having four rows and eight columns according to a preferred embodiment of the invention;

Fig. 6 illustrates how a line draw technique may be used to enlarge an image one row or column at a time for the embodiment discussed above with reference to Fig. 3;

Fig. 7 illustrates how a line draw technique may be used to enlarge an image one row or column at a time for the preferred embodiment discussed above with reference to Fig. 5;

Fig. 8 is a flowchart illustrating how an image is magnified;

Fig. 9 is an illustration of an image having four rows and eight columns being shrunk to an image having three rows and five columns according to one embodiment of the invention;

Fig. 10 illustrates an shrinking of a row of the image of Fig. 9;

Fig. 11 is an illustration of an image having four rows and eight columns being shrunk to an image having three rows and five columns according to a preferred embodiment of the invention;

Fig. 12 illustrates how a line draw technique may be used to shrink an image one row or column at a time for the embodiment discussed above with reference to Fig. 9;

Fig. 13 illustrates how a line draw technique may be used to shrink an image one row or column at a time

for the preferred embodiment discussed above with reference to Fig. 11; and Fig. 14 is a flowchart illustrating how an image is shrunk.

## Detailed Description of the Invention

Fig. 1 is a block diagram of a typical digital computer 100 utilised by a preferred embodiment of the invention. The computer includes main processor(s) 110 coupled to a memory 120, input device(s) 130 and output device(s) 140. Main processor(s) 110 may include a single processor or multiple processors. Input device(s) 130 may include a keyboard, mouse, tablet or other types of input devices. Output device(s) 140 may include a text monitor, plotter or other types of output devices. The main processor may also be coupled to graphics output device(s) 150 such as a graphics display through a graphics adapter 200. Graphics adapter 200 receives instructions regarding graphics from main processor 110 on bus 160. The graphics adapter then executes those instructions with graphics adapter processor(s) 220 coupled to a graphics adapter memory 230. The graphics processors in the graphics adapter then execute those instructions and updates frame buffer(s) 240 based on those instructions. Graphic processors 220 may also include specialised rendering hardware for rendering specific types of primitives to be rendered. Frame buffer(s) 240 includes data for every pixel to be displayed on the graphics output device. A RAMDAC (random access memory digital-to-analog converter) 250 converts the digital data stored in the frame buffers into RGB signals to be provided to the graphics display 150 thereby rendering the desired graphics output from the main processor.

Fig. 2 is a block diagram illustrating the layers of code typically utilised by the host computer and graphics adapter to perform graphics functions. An operating system 300 such as UNIX provides the primary control of the host computer. Coupled to the operating system is an operating system kernel 310 which provides the hardware intensive tasks for the operating system. The operating system kernel communicates directly with the host computer microcode 320. The host computer microcode is the primary instruction set executed by the host computer processor. Coupled to the operating system 300 are graphics applications 330 and 332. This graphics application software can include software packages such as Silicon Graphic's GL, IBM's graPHIGS, MIT's PEX, etc. This software provides the primary functions of two dimensional or three dimensional graphics. Graphics applications 330 and 332 are coupled to graphics application API (application program interface) 340 and 342, respectively. The API provides many of the computationally intensive tasks for the graphics application and provides an interface between the application software and software closer to the graphics hardware such as a device driver for the graphics adapter. For example, API 340 and 342 may communicate with a GAI (graphics application interface) 350 and 352, respectively. The GAI provides an interface between the application API and a graphics adapter device driver 370. In some graphics systems, the API also performs the function of the GAI.

The graphics application, API, and GAI are considered by the operating system and the device driver to be a single process. That is, graphics applications 330 and 332, API 340 and 342, and GAI 350 and 352 are considered by operating system 300 and device driver 370 to be processes 360 and 362, respectively. The processes are identified by the operating system and the device driver by a process identifier (PID) that is assigned to the process by the operating system kernel. Processes 360 and 362 may use the same code that is being executed twice simultaneously, such as two executions of a program in two separate windows. The PID is used to distinguish the separate executions of the same code.

The device driver is a graphics kernel which is an extension of the operating system kernel 310. The graphics kernel communicates directly with microcode of the graphics adapter 380. In many graphics systems, the GAI, or the API if no GAI layer is used, may request direct access from the GAI or API to the adapter microcode by sending an initial request instruction to the device driver. In addition, many graphics systems also allow the adapter microcode to request direct access from the adapter microcode to the GAI or API if no GAI is used by sending an initial request instruction to the device driver. Both processes will hereinafter be referred to as direct memory access (DMA). DMA is typically used when transferring large blocks of data. DMA provides for a quicker transmission of data between the host computer and the adapter by eliminating the need to go through the display driver other than the initial request for the device driver to set up the DMA. In some cases, the adapter microcode utilises context switching which allows the adapter microcode to replace the current attributes being utilised by the adapter microcode. Context switching is used when the adapter microcode is to receive an instruction from a graphics application that utilises different attributes than the adapted microcode is currently using. The context switch is typically initiated by the device driver which recognises the attribute changes.

Blocks 300-342 are software code layers that are typically independent of the type of graphics adapter being utilised. Blocks 350, 352, 370, 380 are software code layers that are typically dependent upon the type of graphics adapter being utilised. For example, if a different graphics adapter were to be used by the graphics application software, then a new GAI 350, 352 graphics kernel 370 and adapter microcode 380 would be need-

ed. In addition, blocks 300-370 reside on and are executed by the host computer. However, the adapter microcode 380 resides on and is executed by the graphics adapter. However, in some cases, the adapter microcode is loaded into the graphics adapter by the host computer during initialisation of the graphics adapter.

In typical graphics systems, the user instructs the graphics application to construct an image from a two or three dimensional model. The user first selects the location and type of light sources. The user then instructs the application software to build the desired model from a set of predefined or user defined objects. Each object may include one or more drawing primitives describing the object. For example, a set of drawing primitives such as many triangles may be used to define the surface of an object. The user then provides a perspective in a window to view the model, thereby defining the desired image. The application software then starts the rendering of the image from the model by sending the drawing primitives describing the objects to the adapter microcode through the API, the GAI, and then the device driver unless DMA is used. The adapter microcode then renders the image on the graphics display by clipping (i.e. not using) those drawing primitives not visible in the window. The adapter microcode then breaks each remaining drawing primitive into visible pixels from the perspective given by the user. The pixels are then loaded into the frame buffer, often with the use of a depth buffer in the case of a three dimensional model. This step is very computationally intensive due to the number of drawing primitives, variables, and pixels involved. The resulting image stored in the frame buffer and displayed on the graphics display typically does not carry the original information such as which drawing primitive or object the pixel was derived from. As a result, the image may need to be rerendered in part or in whole if the window, the user perspective, the model, the lighting, etc. are modified. This also presents a difficulty when trying to zoom or shrink a displayed image in the window. In a typical three dimensional model, when a user has an image displayed in a window, a complicated and sometimes time consuming process is followed.

In the preferred embodiment, the zoom and shrink techniques will be utilised in hardware in the graphics adapter processor. This approach is extremely quick but would probably necessitate specialised hardware. This would allow for rapid zooms or rapid shrinks of observed images displayed by the graphics adapter. The zoom and shrink techniques could also be utilised in many other locations such as the adapter microcode which is close to the adapter frame buffer. This approach would also be relatively quick and fairly easy to implement. In addition, the zoom and shrink techniques would be applied in the graphics application software wherein the rendered image is also stored in system memory either prior to the image being rendered or subsequently by the graphics adapter passing the data back up to the graphics application software. This approach would be much slower but would allow for utilisation of this technique on preexisting graphics adapters. As would be obvious to one of ordinary skill in the art, the present technique would be applied in many other locations within the host computer or graphics adapter.

#### Variable expansion of an image

Fig. 3 is an illustration of an image having three rows and five columns being enlarged to an image having four rows and eight columns according to a first embodiment of the invention. Horizontal lines of pixels called rows may alternatively be defined as columns and vertical lines of pixels called columns may alternatively be defined as rows. Original image 500 is shown with a variable value for each pixel of the image. That is, the pixel on row 1 column 1 has a value of A which may be a grey scale value, an RGB value, or other type of value representing the portion of the image rendered by that pixel.

To perform the zoom, original image 500 is expanded widthwise into calculated image 510. Calculated image 510 is subsequently expanded heightwise into new rendered image 520. In the alternative, rendered image 500 may be first expanded heightwise into calculated image 530 and subsequently expanded widthwise into new rendered image 520.

The image is expanded widthwise or heightwise one row or column at a time. The expansion of a row can be understood, as shown in Fig. 4, by imagining the first row of pixels on the original image 500 being placed on a rubber band which is then stretched and placed over the first row of the calculated image 510.

The end (columns 1 and 5) pixels of row 1 map directly onto the end (columns 1 and 8) pixels of the calculated image and pass their respective pixel values to the calculated image pixels. In addition, the pixels in columns 2 and 4 of original image 500 map well to the pixels in columns 3 and 6 of calculated image 510 and pass their respective pixel values to the calculated image pixels. The pixel in column 3 of original image 500 maps well to columns 4 and 5 of calculated image 510. The pixels in columns 2 and 7 of calculated image 510 are chosen from pixels 2 and 4 of original image 500. Preferably, the pixels are distributed from the original image to the calculated image from left to right in sequential order as will be described in greater detail below with reference to Fig. 6. As a result of this distribution of pixels from the original image to the calculated image, an excellent and quick enlargement of the pixel values is performed for the row.

In subsequent steps, the remaining rows of the original image are expanded into the remaining rows of calculated image 510. After the widthwise expansion has been completed, the calculated image 510 is then enlarged heightwise into the new rendered image 520 one column at a time, similar to the widthwise expansion of each row, into new image 520.

Fig. 5 is an illustration of an image having three rows and five columns being enlarged to an image having four rows and eight columns according to a preferred second embodiment of the invention. Original image 600 is shown with a variable value for each pixel of the image. That is, the pixel on row 1 column 1 has a value of A which may be a grey scale value, an RGB value, or other type of value representing the portion of the image rendered by that pixel.

As described above with reference to Fig. 3, to perform the zoom, original image 600 is expanded widthwise into calculated image 610. Calculated image 610 is subsequently expanded heightwise into new rendered image 620. In the alternative, rendered image 600 may be first expanded heightwise into calculated image 630 and subsequently expanded widthwise into new rendered image 620.

The expansion of row 1 from the original image 600 to calculate image 610 is as follows. The end (columns 1 and 5) pixels of row 1 map directly onto the end (columns 1 and 8) pixels of the calculated image and pass or distribute their respective pixel values to the calculated image pixels. In addition, the pixels in columns 2 and 4 of original image 600 map well to the pixels in columns 3 and 6 of calculated image 610 and pass their respective pixel values to the calculated image pixels. The pixel in column 3 of original image 600 maps between columns 4 and 5 of calculated image 610. The pixels in columns 2, 4, 5 and 7 of calculated image 610 are then interpolated. As can be seen in the illustration, the pixel value in column 2 is interpolated as being a weighted average of the pixel values in columns 1 and 3. In addition, the pixel value in column 7 is interpolated as being a weighted average of the pixel values in columns 6 and 8. Furthermore, the pixels in columns 4 and 5 are interpolated as being a weighted average of the pixel values of columns 3, 4, 5 and 6 of the original image. Preferably, the pixels are distributed from the original image to the calculated image from left to right in sequential order as will be described in greater detail below with reference to Fig. 7. As a result of this distribution of pixels from the original image to the calculated image and the interpolation of values, an excellent enlargement of the pixel values is performed.

As described above with reference to Fig. 3, in subsequent steps, the remaining rows of the original image are expanded into the remaining rows of calculated image 610. After the widthwise expansion has been completed, the calculated image 610 is then enlarged heightwise into the new rendered image 620 one column at a time, similar to the widthwise expansion of each row, into new image 620.

Fig. 6 illustrates how a line draw technique may be used to enlarge an image one row or column at a time for the first embodiment discussed above with reference to Fig. 3. Current line draw techniques are typically used to efficiently calculate how to draw a line in a two dimensional plane without using floating point arithmetic. However, they can be used to magnify images one row or column at a time. This process of magnifying images using a line draw algorithm for each row or column, one at a time, is quick and if an appropriate line draw algorithm is used, then floating point arithmetic may not be necessary to magnify an image.

In the illustrated example given, the five pixel wide row from the original image is the Y axis and the eight pixel wide row from the calculated image is the X axis and the line draw technique is used to determine where the values of the pixels of the original image row are to be allocated to the pixels of the calculated image row. An ideal line 700 is given for drawing a line from a beginning point 710 at the intersection of Column 1 of the original image and the calculated image to an endpoint 720 at the intersection of Column 5 of the original image and Column 8 of the calculated image. However, since discrete elements (in this case pixels) are used, the ideal line cannot be represented. Therefore, an approximation line 730 is used to approximately represent the ideal line. Many techniques exist to create such an approximation line. For example, a Bresenham algorithm is may be used that has the advantage being very quick in many computers because it does not use floating point arithmetic.

The approximation line includes several spans 740A - 740E that are calculated sequentially from the starting point 710 to ending point 720. In this embodiment, each span represents which pixel value of the original image is to be allocated to which pixel value of the calculated image. For example, Column 4 of the calculated image is below span 740C as is shown by intersection point 750. Span 740C represents the pixel value in Column 3 of the original image. Therefore, Column 4 of the calculated image obtains the pixel value C as shown. Each of the pixel values are to be determined in sequential order from Column 1 through Column 8 in the present example. That is, once Column 4 of the calculated image is determined, then Column 5 of the calculated image is determined, and so forth in sequential order.

This process of allocating pixels using a line draw algorithm for each row or column, one at a time, is quick and if an appropriate line draw algorithm is used, then floating point arithmetic may not be necessary to magnify an image.

Fig. 7 illustrates how a line draw technique may be used to enlarge an image one row or column at a time for the preferred second embodiment discussed above with reference to Fig. 5. The preferred embodiment also utilises interpolation of pixel values to provide a more accurate magnification of an image. This interpolation can also be accomplished utilising a modified line draw technique. This process of magnifying images using a line draw algorithm for each row or column, one at a time, is quick and if an appropriate line draw algorithm is used, then floating point arithmetic may not be necessary to magnify an image even if interpolation is used.

As described above with reference to Fig. 6, the five pixel wide row from the original image is the Y axis and the eight pixel wide row from the calculated image is the X axis and the line draw technique is used to determine where the values of the pixels of the original image row are to be allocated to the pixels of the calculated image row and where the values of the pixels in the calculated image are to be interpolated. An ideal line 700 is given for drawing a line from a beginning point 710 at the intersection of Column 1 of the original image and the calculated image to an endpoint 720 at the intersection of Column 5 of the original image and Column 8 of the calculated image. An approximation line 730 is used to approximately represent the ideal line.

The approximation line includes several spans 740A - 740E that are calculated sequentially from the starting point 710 to ending point 720. Each span is utilised to determine which pixel values of the original image is to be allocated or interpolated to the pixel value of the calculated image. For example, Column 4 of the calculated image is below span 740C representing the pixel value in Column 3 of the original image as is shown by intersection point 750. However, Column 4 of the calculated image is also close to span 740B representing Column 2 of the original image. Therefore, Column 4 of the calculated image obtains an interpolated pixel value of  $(B+2C)/3$  as shown based on its proximity to the span representing Column 2 of the original image. This value has been determined from the fact that intersection point 750 is located twice as close to the center of span 740C as it is to the center of span 740B. Each of the pixel values are to be determined in sequential order from Column 1 through Column 8 in the present example. That is, once Column 4 of the calculated image is determined, then Column 5 of the calculated image is determined, and so forth in sequential order.

This process of allocating and interpolating pixels using a line draw algorithm for each row or column, one at a time, is quick and if an appropriate line draw algorithm is used, then floating point arithmetic may not be necessary to magnify an image. A pseudocode routine implementing the preferred second embodiment of the invention is described below with reference to Tables 1A-1D.

Fig. 8 is a flowchart illustrating how an image is magnified. It is noted that the technique is repetitive and lends itself to being performed by a line draw method. In a first step 800, it is determined whether there is a dimension to the image that needs to be magnified. That is, the flowchart is executed only if the image is to be magnified in at least one dimension and may be magnified in many dimensions, such as a three dimensional image that is described by a three dimensional array of data. Assuming there is a dimension remaining to be magnified, then in step 810, the next dimension of magnification is begun by incrementing a counter or calling a subroutine or the like. In step 820, it is determined whether this is the last row or column of data to be magnified for the current dimension being magnified. Given that this is the first execution of this step, the answer would be no and the next (first in this case) row or column of data is begun in step 830. In step 840, it is determined whether this is the last span of pixels to be magnified for the current row or column of data. Since this is the first execution of this step, the answer would be no. If yes though, execution would return to step 820 to determine whether more rows or columns of data need to be magnified. In step 850, the next span is begun by determining which pixels of the magnified image correlate to a pixel of the original image. That is, which pixels of the magnified image are closer to the pixel of the original image represented by this span than the pixels represented by the previous or next spans. In step 860, it is determined whether this is the last pixel of the magnified image for the present span. If yes, then execution returns to step 840 to determine whether this is the last span of the present row or column. If no, then the next, first in this case, pixel is begun in step 870.

In step 900, it is determined whether the current pixel of the magnified image is to be interpolated or simply receive the value of the correlating original image pixel. This determination may be based on various criteria such as proximity to the end of the span and the variation of values for neighboring original image pixels. If the pixel is not to be interpolated, then the pixel receives the value of the correlating original image pixel in step 910 and execution returns to step 860. Otherwise, the pixel receives an interpolated value in step 920 from the correlating original image pixel and surrounding original pixels or surrounding magnified image pixels. Interpolation may be based on various criteria such as proximity of surrounding original image pixels. Once completed, execution returns to step 860.

In the preferred embodiment, a zoom routine is called twice to expand an image in two dimensions, once to expand the image horizontally and once to expand the image vertically. With the proper use of variables, the same routine may be used for both operations. Tables 1A-1D provide a pseudocode routine for expanding an image either horizontally or vertically depending upon the values of the variables passed to the

routine. This routine has been split into four tables to allow better explanation of its functionality.

Table 1A is part of a pseudocode routine for initialising variables in preparation for performing the zoom or magnification operation. The procedure receives the following five variables: AXIS, IMAGE, IN\_SIZE, OUT\_SIZE and NR\_STEPS. AXIS is either the value HORIZ or VERT indicating the direction of the zoom to be performed. That is, a value of HORIZ would indicate that the width of the image is being increased and a value of VERT would indicate that the height of the image is being increased. IMAGE is an array containing the original pixel data prior to the zoom. If the value of AXIS is HORIZ, then IN\_SIZE provides the original width of the pixel array, OUT\_SIZE provides the new width of the pixel array after the zoom, and NR\_STEPS provides the height of the pixel array. If the value of AXIS is VERT, then IN\_SIZE provides the original height of the pixel array, OUT\_SIZE provides the new height of the pixel array and NR\_STEPS provides the width of the pixel array. Space is then allocated in memory for an array called NEW\_IMAGE which is to contain the pixels of the zoomed image upon completion of the zoom routine. The size of NEW\_IMAGE is equal to NR\_STEPS times OUT\_SIZE.

Several constants necessary for performing the zoom are then calculated. SPAN\_A is the first middle span length and is based on the amount the image is to be expanded. For example, an image being expanded from 4 to 9 pixels in width would have a SPAN\_A of 3. SPAN\_B is the second middle run length and is one less than SPAN\_A. SPAN\_A and SPAN\_B are chosen that they are the nearest integer values above and below, respectively, the inverse of the slope of the line to be approximated. SPAN\_INIT is the initial run length and is one-half of SPAN\_A. SPAN\_FINAL is the final run length. SPAN\_INIT and SPAN\_FINAL represent the portion left over at the beginning and end of the line to be approximated.

ERROR is equal to the error accumulated during the magnification process. POS\_DELTA is equal to positive error correction and NEG\_DELTA is equal to the negative error correction. N and R are intermediate values used for calculations. SCAN\_INC is an offset value that provides the number of entries needed to get to the next entry on the same row in the array.

Table 1A: Initialise Zoom Variables

```

5      /* Copyright International Business Machines      */
      /*      Corporation, 1992 All Rights Reserved      */
      /*                                                    */
10     /* procedure zoom: receive zoom variables          */
      ZOOM (AXIS,IMAGE,IN_SIZE,OUT_SIZE,NR_STEPS)
      /* allocate space to store the zoomed image        */
      NEW_IMAGE = Allocate (NR_STEPS * OUT_SIZE)
15     /* compute constants                                */
      SPAN_A = Integer (OUT_SIZE / IN_SIZE + .99999)
      SPAN_B = SPAN_A - 1
      SPAN_INIT = Integer (SPAN_A / 2)
20     R = OUT_SIZE - SPAN_A * IN_SIZE
      If (SPAN_A is ODD)
          then N = R + IN_SIZE
25         else N = R
      If (N = 0)
          then SPAN_FINAL = SPAN_INIT - 1
          else SPAN_FINAL = SPAN_INIT
30     NEG_DELTA = 2 * (R - IN_SIZE)
      POS_DELTA = 2 * R
      SPAN_A = SPAN_A + 1
      SPAN_B = SPAN_B + 1
35     If (AXIS = HORIZ)
          then SCAN_INC = 1
      If (AXIS = VERT)
40         then SCAN_INC = NR_STEPS
      /* continue to table 1B                                */

```

Table 1B is a pseudocode routine for initiating the outer loop of the zoom routine. The outer loop is performed once for each row or column, depending upon whether the zoom is horizontal or vertical. A row or column of data is then set up in working storage for use in calculations. Some loop variables are initialised for each row or column of pixels to be expanded.

Table 1B: Start Zoom Outer Loop for Row or Column

```

5      /* start outer loop that increments by row or col */
      For I = 0 to NR_STEPS by 1 do
      /* set up row or column of data for zoom */
          If (AXIS = HORIZ)
10             then
                SOURCE_PIX = IMAGE + (I * IN_SIZE)
                DEST_PIX = NEW_IMAGE + (I * OUT_SIZE)
15             else
                SOURCE_PIX = IMAGE + I
                DEST_PIX = NEW_IMAGE + I
      /* initialisation of outer loop variables */
20      LAST_H = 1 + (2 * H_INITIAL)
      AVG_IDX = 0
      LAST_IDX = 1
      LAST_PIX = SOURCE_PIX
25      Copy (DEST_PIX, SOURCE_PIX)
      DEST_PIX = DEST_PIX + SCAN_INC
      SOURCE_PIX = SOURCE_PIX + SCAN_INC
30      ERROR = N + NEG_DELTA
      /* continue to Table 1C */

```

35 Table 1C is a pseudocode routine for initiating the inner loop of the zoom routine. The inner loop is then begun for each span within the outer loop row or column of pixels. Various intermediate values are computed in preparation for linear interpolation of pixel values.

40

45

50

55

Table 1C: Perform Zoom Inner Loop

```

5      /* start inner loop that increments every span      */
      For J = 1 to IN_SIZE do
          If (ERROR < 0)
              then
10              H = SPAN_B
              ERROR = ERROR + POS_DELTA
              else
15              H = SPAN_A
              ERROR = ERROR + NEG_DELTA
              AVG_IDX = AVG_IDX + LAST_H + H
          /* save the value of H for future use      */
20          LAST_H = H
          /* prepare variables to linearly interpolate */
              DELTA_SPAN = AVG_IDX - LAST_IDX
              DELTA_RED = SOURCE_PIX (RED) - LAST_PIX (RED)
25              DELTA_GRN = SOURCE_PIX (GRN) - LAST_PIX (GRN)
              DELTA_BLU = SOURCE_PIX (BLU) - LAST_PIX (BLU)
          /* continue to Table 1D                      */

```

Table 1D is a pseudocode routine for initiating the center loop of the zoom routine. The center loop is begun for each pixel within the span. The pixel values are then calculated. Then the outer loop is returned and the final zoomed image is then returned to the calling program.

Table 1D: Perform Zoom Interpolate

```

5      /* start center loop that increments by pixel      */
      For K = 0 to infinity by 1 while
        (LAST_IDX <= AVG_IDX) do
10      If LAST_IDX is EVEN
        then
          DEST_PIX (RED) = LAST_PIX (RED) +
            ((K * DELTA_RED)/DELTA_SPAN)
15      DEST_PIX (GRN) = LAST_PIX (GRN) +
            ((K * DELTA_GRN)/DELTA_SPAN)
          DEST_PIX (BLU) = LAST_PIX (BLU) +
            ((K * DELTA_BLU)/DELTA_SPAN)
20      /* after filling a pixel, point to the next pixel */
          DEST_PIX = DEST_PIX + SCAN_INC
          LAST_IDX = LAST_IDX + 1
25      Next K
      /* end of center loop                                */
      /* save data for the next pass of the inner loop     */
          LAST_PIX = SOURCE_PIX
30      SOURCE_PIX = SOURCE_PIX + SCAN_INC
      Next J
      /* end of inner loop                                  */
35      Next I
      /* end of outer loop                                  */
      /* return the new array to the caller.                */
      Return (NEW_IMAGE)

```

Although the variable expansion has been fully described above with reference to specific embodiments, other alternative embodiments will be apparent to those of ordinary skill in the art. For example, alternative line draw techniques or methods may be utilised. In addition, alternative interpolation techniques may be utilised.

#### Variable shrinking of an image

Fig. 9 is an illustration of an image having four rows and eight columns being shrunk or minimised into an image having three rows and five columns according to a third embodiment of the invention. Horizontal lines of pixels called rows may alternatively be defined as columns and the vertical lines of pixels called columns may alternatively be defined as rows. The Original image 1500 is shown with a variable value for each pixel of the image. That is, the pixel on row 1 column 1 has a value of A which may be a grey scale value, an RGB value, or other type of value representing the portion of the image rendered by that pixel.

To perform the shrink, original image 1500 is shrunk widthwise into calculated image 1510. Calculated image 1510 is subsequently shrunk heightwise into new rendered image 1520. In the alternative, rendered image 1500 may be first shrunk heightwise into calculated image 1530 and subsequently shrunk widthwise into new rendered image 1520.

The image is shrunk widthwise or heightwise one row or column at a time. The shrinkage of a row can be understood, as shown in Fig. 10, by imagining the first row of pixels on the original image 1500 being placed

on a stretched rubber band which is then relaxed and placed under the first row of the calculated image 1510.

The end (columns 1 and 8) pixels of row 1 map directly onto the end (columns 1 and 5) pixels of the calculated image and pass their respective pixel values to the calculated image pixels. In addition, the pixels in columns 3 and 6 of original image 1500 map well to the pixels in columns 2 and 4 of calculated image 1510 and pass their respective pixel values to the calculated image pixels. The pixels in columns 4 and 5 of original image 1500 both map well to column 3 of calculated image 1510. In this embodiment, the value of the pixel with the lowest column number is selected although the value of the pixel with the highest column number may also be selected. Preferably, the pixels are allocated from the original image to the calculated image from left to right in sequential order as will be described in greater detail below with reference to Fig. 12. As a result of this allocation of pixels from the original image to the calculated image, an excellent and quick shrinkage of the pixel values is performed for the row.

In subsequent steps, the remaining rows of the original image are shrunk into the remaining rows of calculated image 1510. After the widthwise shrinkage has been completed, the calculated image 1510 is then shrunk heightwise into the new rendered image 1520 one column at a time, similar to the widthwise shrinkage of each row, into new image 1520.

Fig. 11 is an illustration of an image having four rows and eight columns being shrunk to an image having three rows and five columns according to a preferred fourth embodiment of the invention. Original image 1600 is shown with a variable value for each pixel of the image. That is, the pixel on row 1 column 1 has a value of A which may be a grey scale value, an RGB value, or other type of value representing the portion of the image rendered by that pixel.

As described above with reference to Fig. 9, to perform the shrink, original image 1600 is shrunk widthwise into calculated image 1610. Calculated image 1610 is subsequently shrunk heightwise into new rendered image 1620. In the alternative, rendered image 1600 may be first shrunk heightwise into calculated image 1630 and subsequently shrunk widthwise into new rendered image 1620.

The shrinking of row 1 from the original image 1600 to calculate image 1610 is as follows. The end (columns 1 and 8) pixels of row 1 map directly onto the end (columns 1 and 5) pixels of the calculated image and pass their respective pixel values to the calculated image pixels. The pixels in columns 2 and 3 of original image 1600 both map to the pixel in columns 2 and 6 of calculated image 1610. The pixels in columns 4 and 5 of original image 1600 both map to the pixel in column 3 of calculated image 1610. The pixels in columns 6 and 7 of original image 1600 both map to the pixel in column 4 of calculated image 1610. The pixels in columns 2, 3, and 4 of calculated image 1610 are then calculated from pixels 2, 3, 4, 5, 6 and 7 of the original image of original image 1600. As can be seen in the illustration, the pixel value in column 2 is calculated as an average of the pixel values in columns 2 and 3. In addition, the pixel value in column 3 is calculated as an average of the pixel values in columns 4 and 5. Furthermore, the pixel value in column 4 is calculated as an average of the pixel values of columns 6 and 7 of the original image. Preferably, the pixels are calculated from the original image to the calculated image from left to right in sequential order as will be described in greater detail below with reference to Fig. 13. As a result of this calculation of pixels from the original image to the calculated image, an excellent shrinking of the pixel values is performed.

As described above with reference to Fig. 9, in subsequent steps, the remaining rows of the original image are shrunk into the remaining rows of calculated image 1610. After the widthwise shrinking has been completed, the calculated image 1610 is then shrunk heightwise into the new rendered image 1620 one column at a time, similar to the widthwise expansion of each row, into new image 1620.

Fig. 12 illustrates how a line draw technique may be used to shrink an image one row or column at a time for the third embodiment discussed above with reference to Fig. 9. Current line draw techniques are typically used to efficiently calculate how to draw a line in a two dimensional plane without using floating point arithmetic. However, they can be used to shrink images one row or column at a time. This process of shrinking images using a line draw algorithm for each row or column, one at a time, is quick and if an appropriate line draw algorithm is used, then floating point arithmetic may not be necessary to shrink an image.

In the illustrated example given, the eight pixel wide row from the original image is the X axis and the five pixel wide row from the calculated image is the Y axis and the line draw technique is used to determine where the values of the pixels of the original image row are to be correlated to the pixels of the calculated image row. An ideal line 1700 is given for drawing a line from a beginning point 1710 at the intersection of Column 1 of the original image and the calculated image to an endpoint 1720 at the intersection of Column 5 of the original image and Column 8 of the calculated image. However, since discrete elements (in this case pixels) are used, the ideal line cannot be represented. Therefore, an approximation line 1730 is used to approximately represent the ideal line. Many techniques exist to create such an approximation line. For example, a Bresenham algorithm may be used that has the advantage being very quick in many computers because it does not use floating point arithmetic.

The approximation line includes several spans 1740A - 1740E that are calculated sequentially from the starting point 1710 to ending point 1720. In this embodiment, each span represents which pixel values of the original image are to be correlated to which pixel value of the calculated image. For example, Column 3 of the calculated image is next to span 1740C as is shown by intersection point 1750. Span 1740C represents the pixel values in Columns 4 and 5 of the original image. Therefore, Column 3 of the calculated image obtains the pixel value D as shown. Each of the pixel values are to be determined in sequential order from Column 1 through Column 5 in the present example. That is, once Column 3 of the calculated image is determined, then Column 4 of the calculated image is determined, and so forth in sequential order.

This process of determining pixel values using a line draw algorithm for each row or column, one at a time, is quick and if an appropriate line draw algorithm is used, then floating point arithmetic may not be necessary to shrink an image.

Fig. 13 illustrates how a line draw technique may be used to shrink an image one row or column at a time for the preferred fourth embodiment discussed above with reference to Fig. 11. The preferred embodiment also utilises calculation of pixel values to provide a more accurate shrinking of an image. This calculation can also be accomplished utilising a modified line draw technique. This process of shrinking images using a line draw algorithm for each row or column, one at a time, is quick and if an appropriate line draw algorithm is used, then floating point arithmetic may not be necessary to shrink an image.

As described above with reference to Fig. 12, the eight pixel wide row from the original image is the X axis and the five pixel wide row from the calculated image is the Y axis and the line draw technique is used to determine where the values of the pixels of the original image row are to be used to determine the pixel values of the calculated image row. An ideal line 1700 is given for drawing a line from a beginning point 1710 at the intersection of Column 1 of the original image and the calculated image to an endpoint 1720 at the intersection of Column 8 of the original image and Column 5 of the calculated image. An approximation line 1730 is used to approximately represent the ideal line.

The approximation line includes several spans 1740A - 1740E that are calculated sequentially from the starting point 1710 to ending point 1720. Each span is utilised to determine which pixel values of the original image is to be used to calculate the pixel value of the calculated image. For example, Column 3 of the calculated image is next to span 1740C representing the pixel values in Columns 4 and 5 of the original image as is shown by intersection point 1750. Therefore, Column 4 of the calculated image obtains an calculated weighted average pixel value of  $(D+E)/2$  as shown. Each of the pixel values are to be determined in sequential order from Column 1 through Column 5 in the present example. That is, once Column 3 of the calculated image is determined, then Column 4 of the calculated image is determined, and so forth in sequential order.

This process of calculating pixels using a line draw algorithm for each row or column, one at a time, is quick and if an appropriate line draw algorithm is used, then floating point arithmetic may not be necessary to shrink an image. A pseudocode routine implementing the preferred embodiment of the invention is described below with reference to Tables 2A-2B.

Fig. 14 is a flowchart illustrating how an image is shrunk or minimised. It is noted that the technique is repetitive and lends itself to being performed by a line draw method. In a first step 1800, it is determined whether there is a dimension to the image that needs to be shrunk. That is, the flowchart is executed only if the image is to be shrunk in at least one dimension and may be shrunk in many dimensions, such as a three dimensional image that is described by a three dimensional array of data. Assuming there is a dimension remaining to be shrunk, then in step 1810, the next dimension of shrinkage is begun by incrementing a counter or calling a subroutine or the like. In step 1820, it is determined whether this is the last row or column of data to be shrunk for the current dimension being shrunk. Given that this is the first execution of this step, the answer would be no and the next (first in this case) row or column of data is begun in step 1830. In step 1860, it is determined whether this is the last pixel to be shrunk for the current row or column of data. Since this is the first execution of this step, the answer would be no. If yes though, execution would return to step 1820 to determine whether more rows or columns of data need to be shrunk. In step 1870, the next pixel is begun by determining which pixels of the original image correlate to a pixel of the shrunk image. That is, which pixels of the original image are closer to the pixel of the shrunk image than the previous or next pixels. In step 1880, the current pixel value of the shrunk image is calculated based on the correlated pixel values and execution is returned to step 1860.

In the preferred embodiment, a shrink routine is called twice to shrink an image in two dimensions, once to shrink the image horizontally and once to shrink the image vertically. With the proper use of variables, the same routine may be used for both operations. Tables 2A-2B provide a pseudocode routine for shrinking an image either horizontally or vertically depending upon the values of the variables passed to the routine. This routine has been split into two tables to allow better explanation of its functionality.

Table 2A is part of a pseudocode routine for initialising variables in preparation for performing the shrink

or minimisation operation. The procedure receives the following five variables: AXIS, IMAGE, IN\_SIZE, OUT\_SIZE and NR\_STEPS. AXIS is either the value HORIZ or VERT indicating the direction of the shrink to be performed. That is, a value of HORIZ would indicate that the width of the image is being decreased and a value of VERT would indicate that the height of the image is being decreased. IMAGE is an array containing the original pixel data prior to the shrink. If the value of AXIS is HORIZ, then IN\_SIZE provides the original width of the pixel array, OUT\_SIZE provides the new width of the pixel array after the shrink, and NR\_STEPS provides the height of the pixel array. If the value of AXIS is VERT, then IN\_SIZE provides the original height of the pixel array, OUT\_SIZE provides the new height of the pixel array and NR\_STEPS provides the width of the pixel array. Space is then allocated in memory for an array called NEW\_IMAGE which is to contain the pixels of the shrunken image upon completion of the shrink routine. The size of NEW\_IMAGE is equal to NR\_STEPS times OUT\_SIZE.

Several constants necessary for performing the zoom are then calculated. ERROR is equal to the error accumulated during the shrinking process. POS\_DELTA is equal to positive error correction and NEG\_DELTA is equal to the negative error correction. N and R are intermediate values used for calculations. SCAN\_INC is an offset value that provides the number of entries needed to get to the next entry on the same row in the array.

The outer loop of the shrink routine is then initiated. The outer loop is performed once for each row or column, depending upon whether the shrink is horizontal or vertical. A row or column of data is then set up in working storage for use in calculations. Some loop variables are initialised for each row or column of pixels to be expanded.

Table 2A: Initialise Shrink Variables

```

5      /* Copyright International Business Machines      */
      /*      Corporation, 1992 All Rights Reserved      */
      /*                                                              */
      /* procedure shrink: receive shrink variables      */
10     Shrink(AXIS, IMAGE, IN_SIZE, OUT_SIZE, NR_STEPS)
      /* allocate space to store the zoomed image      */
      NEW_IMAGE = Allocate (NR_STEPS * OUT_SIZE)
15     /* compute constants      */
      POS_DELTA = 2 * OUT_SIZE
      NEG_DELTA = POS_DELTA - (2 * IN_SIZE)
      ERROR = POS_DELTA - IN_SIZE
20     If (AXIS = HORIZ) then SCAN_INC = 1
      If (AXIS = VERT) then SCAN_INC = NR_STEPS
      /* begin outer loop      */
      For I = 0 to NR_STEPS by 1 do
25         If (AXIS = HORIZ) then do
            SOURCE_PIX = IMAGE + I * IN_SIZE
            DEST_PIX = NEW_IMAGE + I * OUT_SIZE
30         else do
            SOURCE_PIX = IMAGE + I
            DEST_PIX = NEW_IMAGE + I
            RED_SUM = SOURCE_PIX->RED
35            GRN_SUM = SOURCE_PIX->GRN
            BLU_SUM = SOURCE_PIX->BLU
            STEP_SUM = 1
40     /* continue to table 2B      */

```

Table 2B is a pseudocode routine for initiating the inner loop of the zoom routine. The inner loop is then begun for each pixel within the outer loop row or column of pixels. The pixel values are then calculated. Then the inner and outer loop are incremented and the final shrunken image is then returned to the calling program.

Table 2B: Perform Shrink Inner Loop

```

5      /* inner loop                                     */
      Do (IN_SIZE) times
        If (ERROR >= 0) then do
          ERROR = ERROR + NEG_DELTA
10      DEST_PIX(RED) = RED_SUM/STEP_SUM
          DEST_PIX(GRN) = GRN_SUM/STEP_SUM
          DEST_PIX(BLU) = BLU_SUM/STEP_SUM
15      DEST_PIX = DEST_PIX + SCAN_INC
          RED_SUM = SOURCE_PIX(RED)
          GRN_SUM = SOURCE_PIX(GRN)
          BLU_SUM = SOURCE_PIX(BLU)
20      STEP_SUM = 1
        else
          ERROR = ERROR + POS_DELTA
          RED_SUM = RED_SUM + SOURCE_PIX(RED)
25      GRN_SUM = GRN_SUM + SOURCE_PIX(GRN)
          BLU_SUM = BLU_SUM + SOURCE_PIX(BLU)
          STEP_SUM = STEP_SUM + 1
30      SOURCE_PIX = SOURCE_PIX + SCAN_INC
      /* end of inner loop                               */
      Next IN_SIZE
      /* end of outer loop                               */
35      Next I
      /* return the new array to the caller.             */
      Return (NEW_IMAGE)

```

40 Although the variable shrinkage has been fully described above with reference to specific embodiments, other alternative embodiments will be apparent to those of ordinary skill in the art. For example, alternative line draw techniques or methods may be utilised. In addition, alternative averaging techniques such as weighting the accumulated original pixels by a quantity inversely proportional to the corresponding error between  
45 the ideal and approximation mapping lines, may be utilised.

### Claims

- 50 1. A method of expanding an image stored in a first array (500) comprising the steps of:  
 separating (820, 830) said first array into a plurality of rows, each row having a first plurality of data elements, each element having a value; and  
 separately expanding each row from a first plurality of data elements in said first array to a second plurality of data elements in a second array (510) including the steps of:  
 55 correlating (850) each of said first plurality of data elements to at least one of said second plurality of data elements; and  
 distributing (910) the value of selected elements of said first plurality of data elements to correlated data elements of said second plurality of data elements.

2. A method as claimed in Claim 1 wherein said step of expanding includes interpolating values for each of the elements of the second plurality of data elements that were not distributed values of selected elements of said first plurality of data elements.
- 5 3. A method as claimed in any preceding claim further comprising the steps of:  
separating said second array (510) into a plurality of columns, each column having a third plurality of data elements, each element having a value; and  
separately expanding each column from a third plurality of data elements in said second array (510) to a  
10 fourth plurality of data elements in a third array (520) including the steps of:  
correlating (850) each of said third plurality of data elements to at least one of said fourth plurality  
of data elements; and  
distributing (910) the value of selected elements of said third plurality of data elements to correlated  
data elements of said fourth plurality of data elements.
- 15 4. A method as claimed in Claim 3 wherein said step of expanding includes interpolating values for each of  
the elements of the second and fourth plurality of data elements that were not distributed values of se-  
lected elements of said first and third plurality of data elements.
- 20 5. A method of shrinking an image stored in a first array (1500) comprising the steps of:  
separating (1820, 1830) said first array into a plurality of rows, each row having a first plurality of data  
elements, each element having a value; and  
separately shrinking each row from a first plurality of data elements in said first array to a second plurality  
of data elements in a second array (1510) including the steps of:  
25 correlating (1880) each of said second plurality of data elements to at least one of said first plurality  
of data elements; and  
calculating (1880) the value of each of said second plurality of data elements from the values of  
correlated data elements.
- 30 6. A method as claimed in Claim 5 further comprising the steps of:  
separating said second array (1510) into a plurality of columns, each column having a third plurality of  
data elements, each element having a value; and  
separately shrinking each column from a third plurality of data elements in said second array (1510) to a  
fourth plurality of data elements in a third array (1520) including the steps of:  
35 correlating (1880) each of said fourth plurality of data elements to at least one of said third plurality  
of data elements; and  
calculating (1880) the value of each of said fourth plurality of data elements from the values of cor-  
related data elements.
- 40 7. A method as claimed in Claim 6 wherein said step (1880) of calculating includes averaging the value of  
each of said second and fourth plurality of data elements from the values of correlated data elements.
8. A method as claimed in claim 6 wherein said step of shrinking includes shrinking without utilising floating  
point arithmetic.
- 45 9. An apparatus for expanding an image stored in a first array (500) comprising:  
means for separating said first array (500) into a plurality of rows, each row having a first plurality of data  
elements, each element having a value; and  
means for separately expanding each row from a first plurality of data elements in said first array to a  
second plurality of data elements in a second array (510) including:  
50 means for correlating each of said first plurality of data elements to at least one of said second  
plurality of data elements; and  
means for distributing the value of selected elements of said first plurality of data elements to cor-  
related data elements of said second plurality of data elements.
- 55 10. An apparatus as claimed in Claim 9 wherein said means for expanding includes means for interpolating  
values for each of the elements of the second plurality of data elements that were not distributed values  
of selected elements of said first plurality of data elements.
11. An apparatus as claimed in either Claim 9 or Claim 10 further comprising:

means for separating said second array (510) into a plurality of columns, each column having a third plurality of data elements, each element having a value; and  
 means for separately expanding each column from a third plurality of data elements in said second array (510) to a fourth plurality of data elements in a third array (520) including:

means for correlating each of said third plurality of data elements to at least one of said fourth plurality of data elements; and

means for distributing the value of selected elements of said third plurality of data elements to correlated data elements of said fourth plurality of data elements.

12. An apparatus as claimed in Claim 11 wherein said means for expanding include means for interpolating values for each of the elements of the second and fourth plurality of data elements that were not distributed values of selected elements of said first and third plurality of data elements.

13. An apparatus for shrinking an image stored in a first array (1500) comprising:

means for separating said first array (1500) into a plurality of rows, each row having a first plurality of data elements, each element having a value; and  
 means for separately shrinking each row from a first plurality of data elements in said first array to a second plurality of data elements in a second array (1510) including:

means for correlating each of said second plurality of data elements to at least one of said first plurality of data elements; and

means for calculating the value of each of said second plurality of data elements from the values of correlated data elements.

14. An apparatus as claimed in Claim 13 further comprising:

means for separating said second array (1510) into a plurality of columns, each column having a third plurality of data elements, each element having a value; and  
 means for separately shrinking each column from a third plurality of data elements in said second array to a fourth plurality of data elements in a third array (1520) including:

means for correlating each of said fourth plurality of data elements to at least one of said third plurality of data elements; and

means for calculating the value of each of said fourth plurality of data elements from the values of correlated data elements.

15. An apparatus as claimed in Claim 14 wherein said means for calculating includes means for averaging the value of each of said second and fourth plurality of data elements from the values of correlated data elements.

16. An apparatus as claimed in Claim 14 wherein said means for shrinking include means for shrinking without utilising floating point arithmetic.

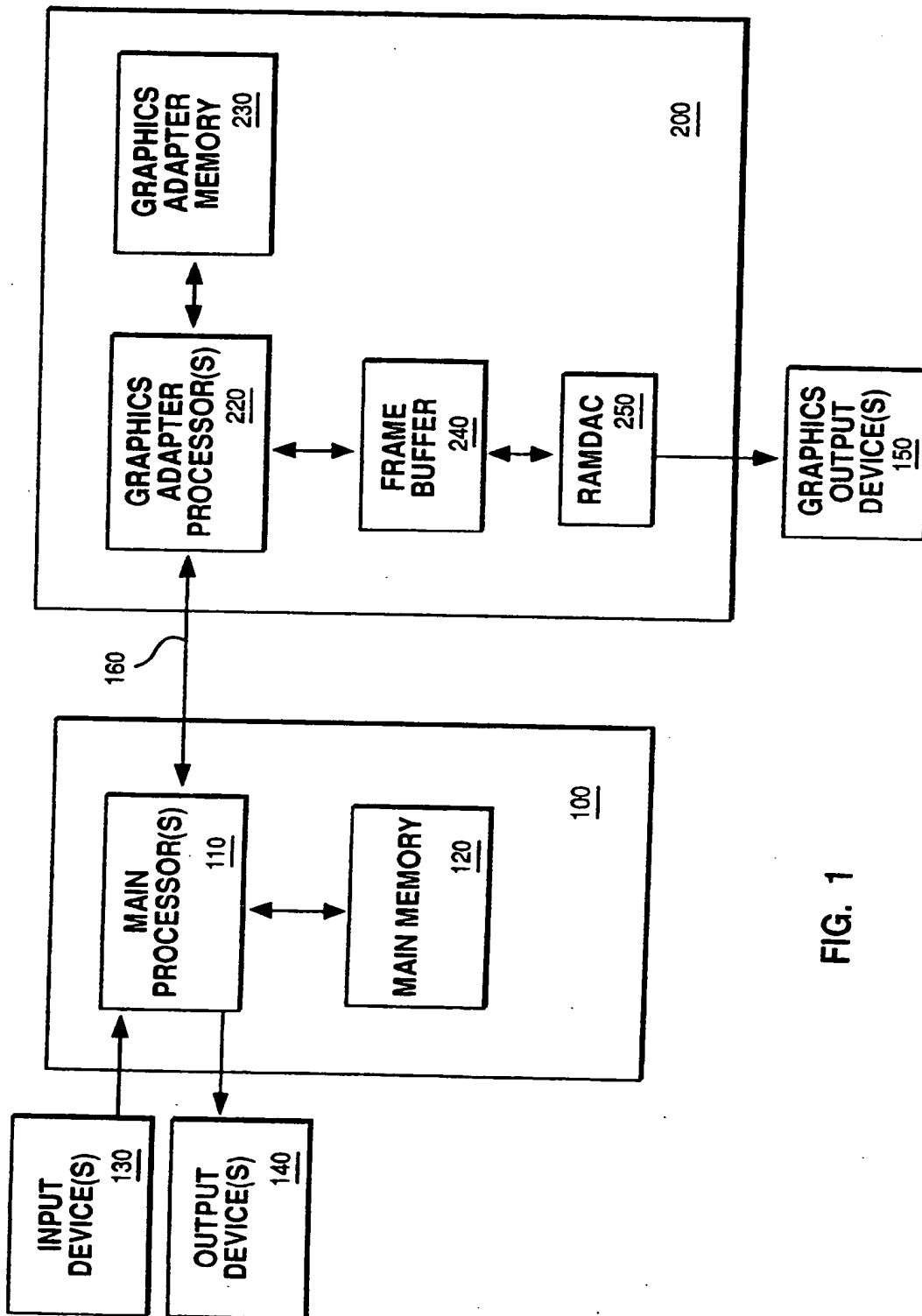


FIG. 1

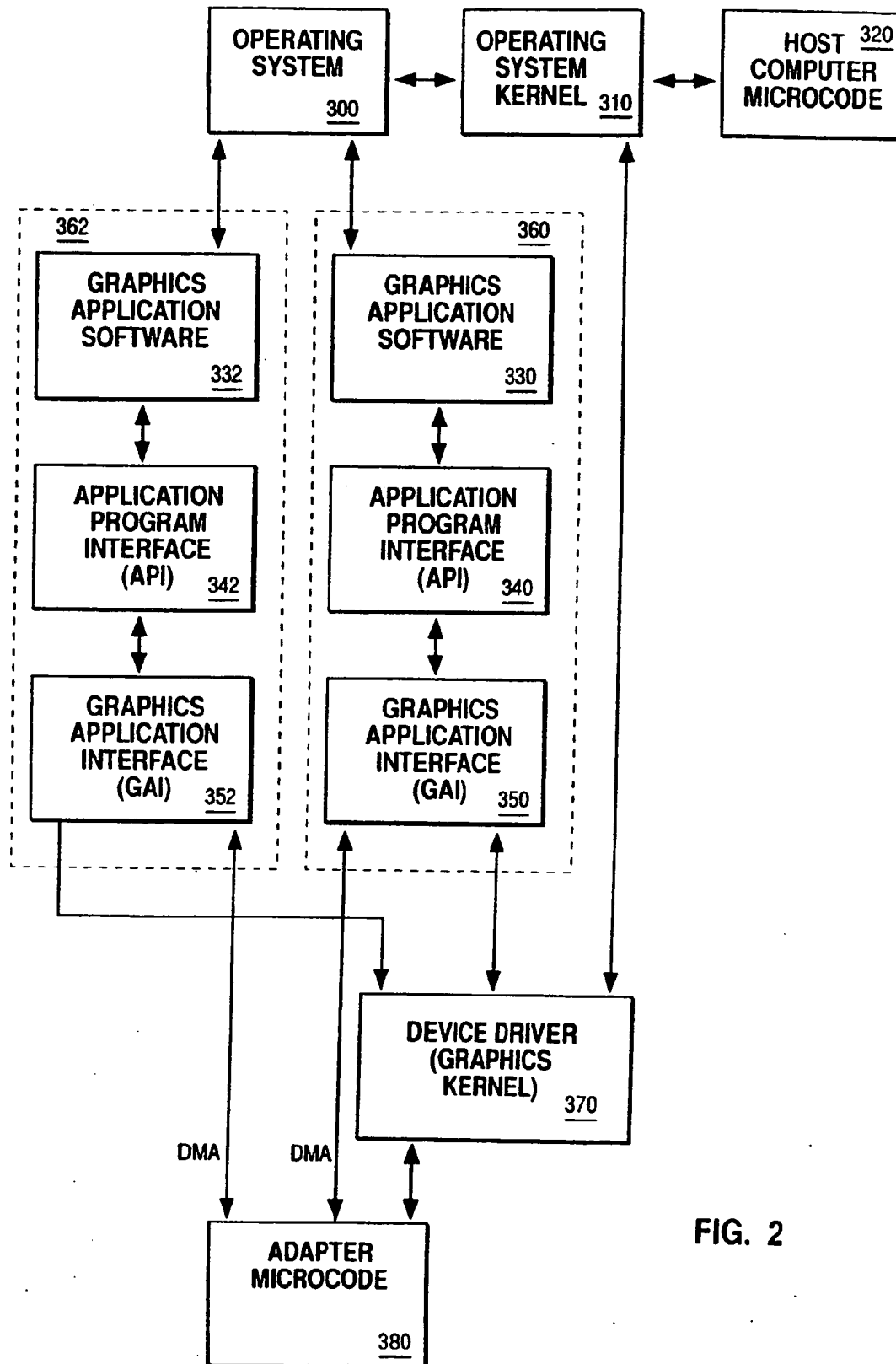


FIG. 2

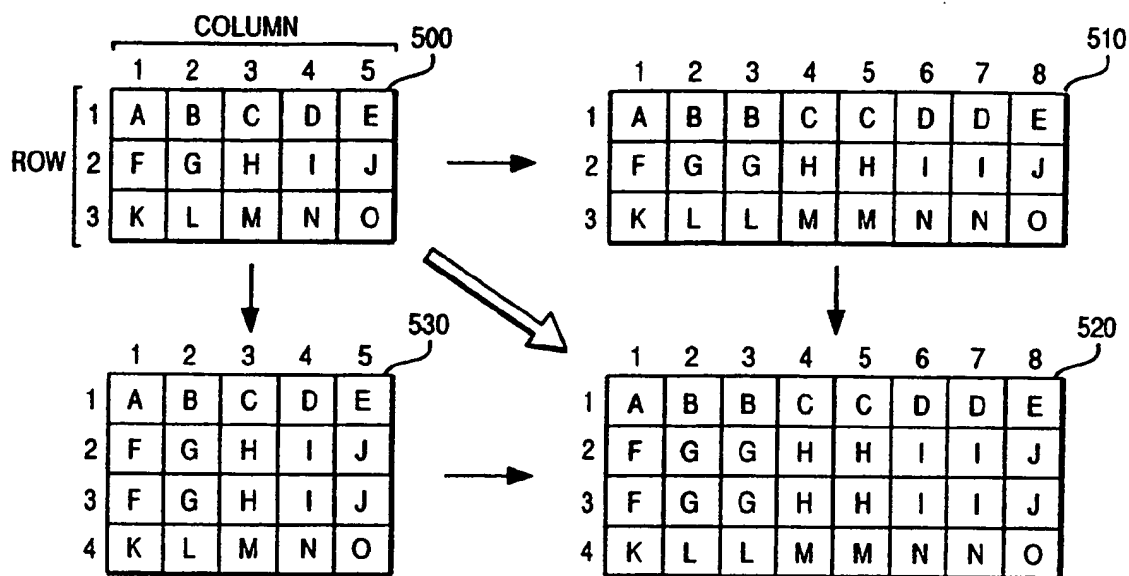


FIG. 3

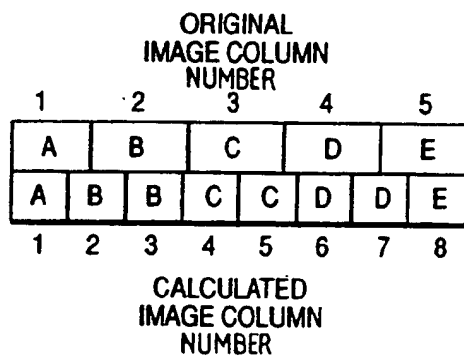


FIG. 4

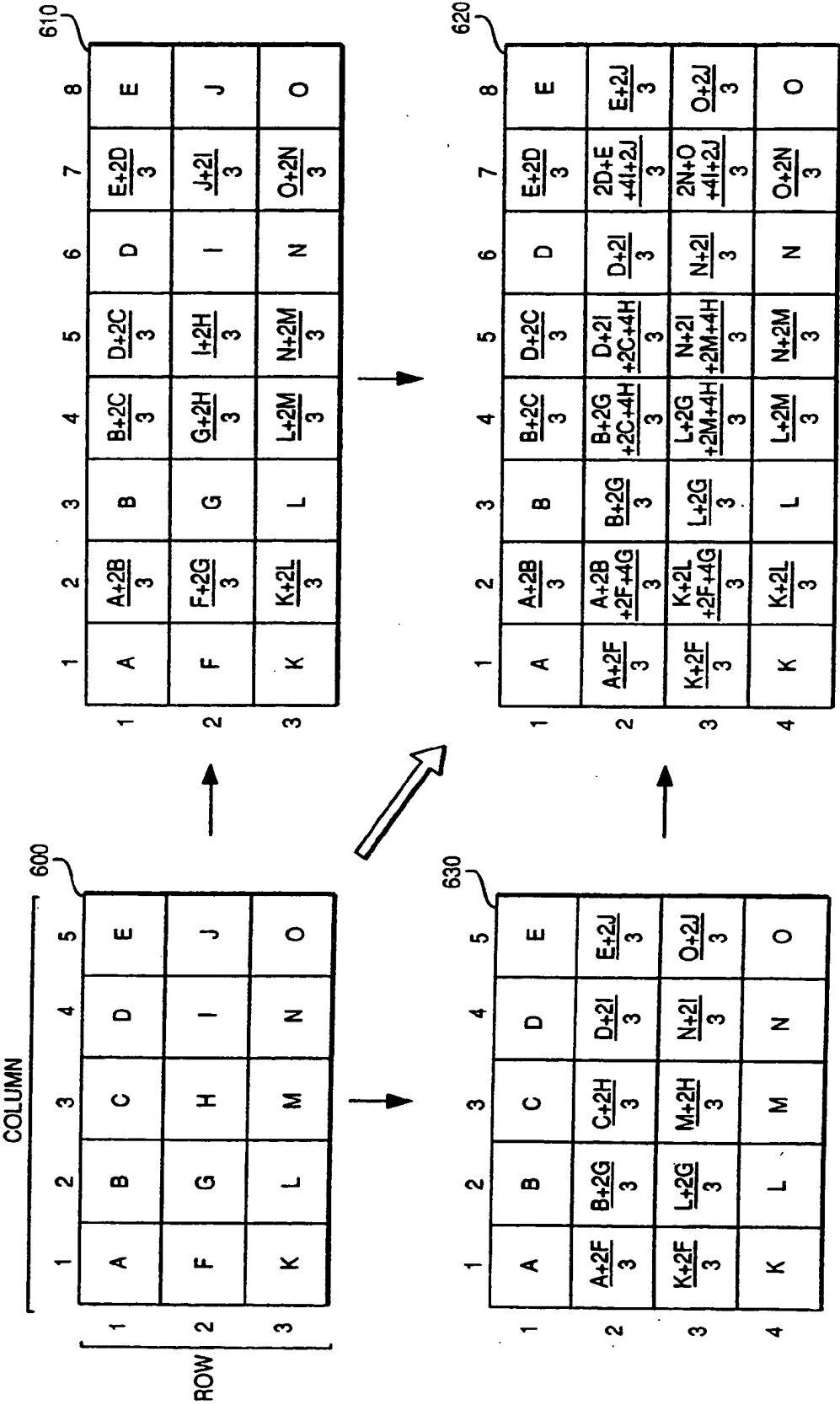


FIG. 5

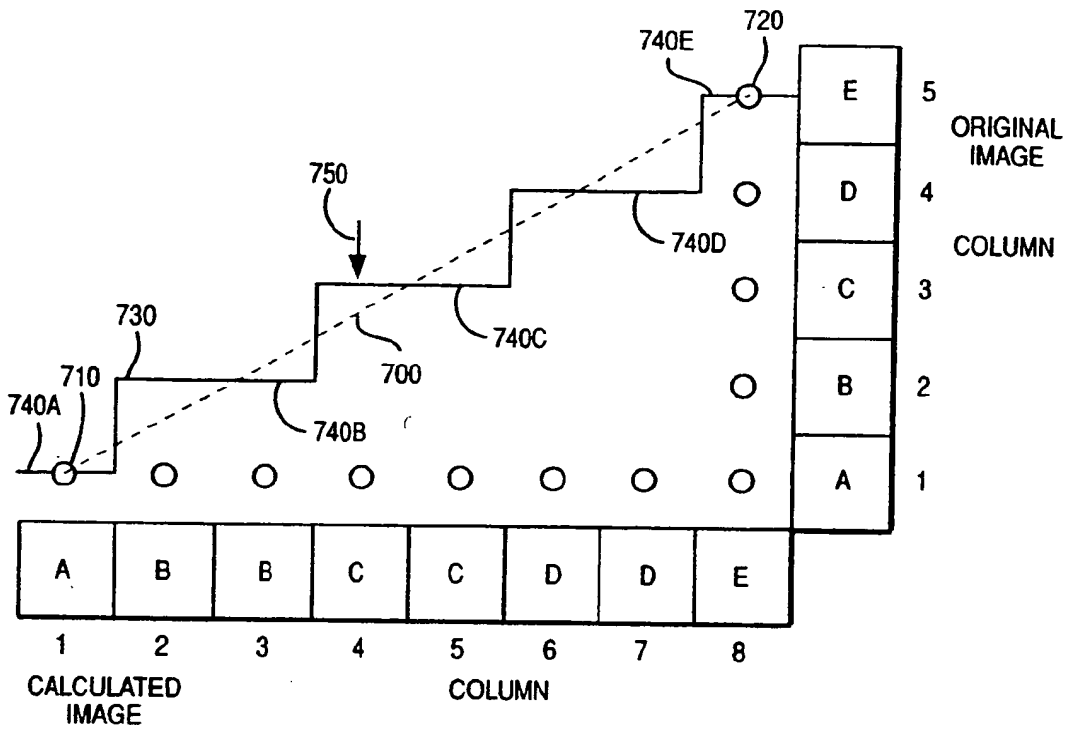


FIG. 6

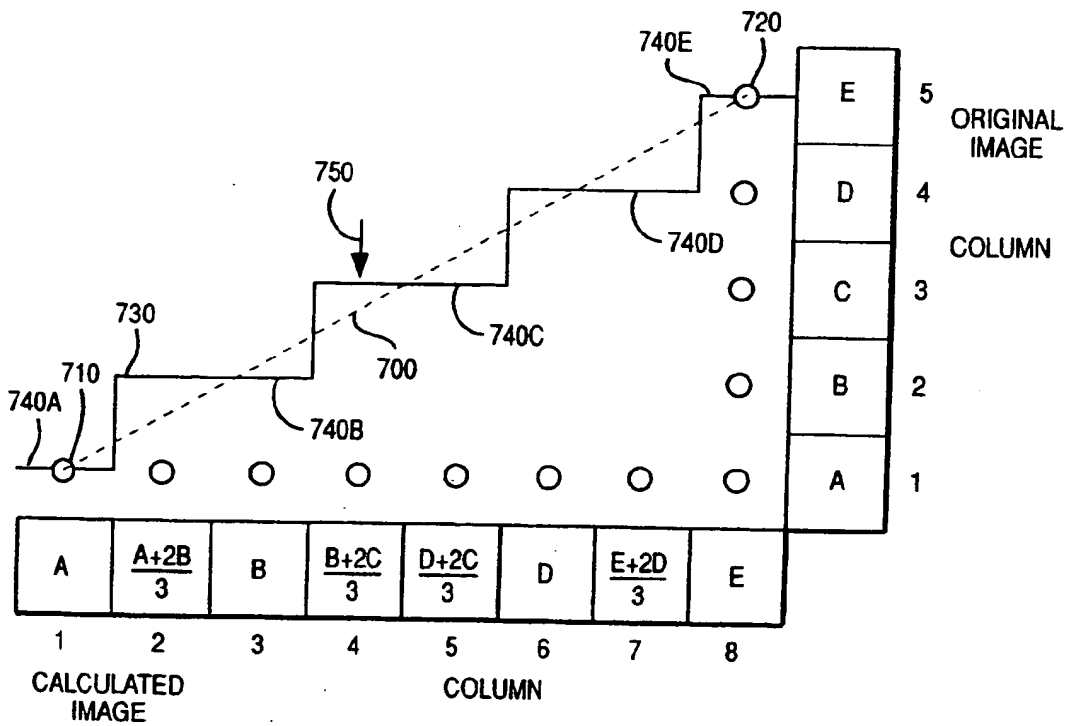


FIG. 7

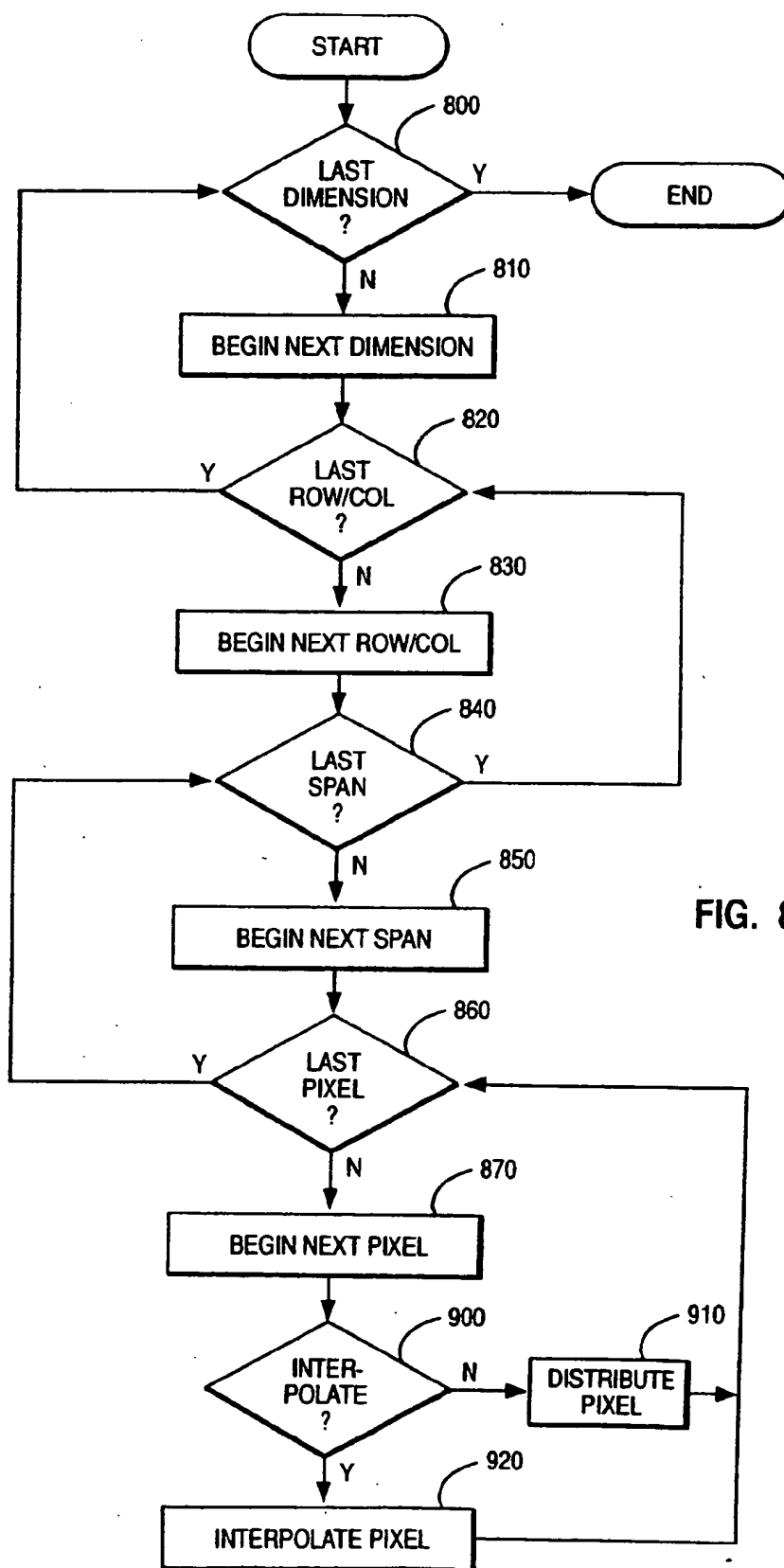


FIG. 8

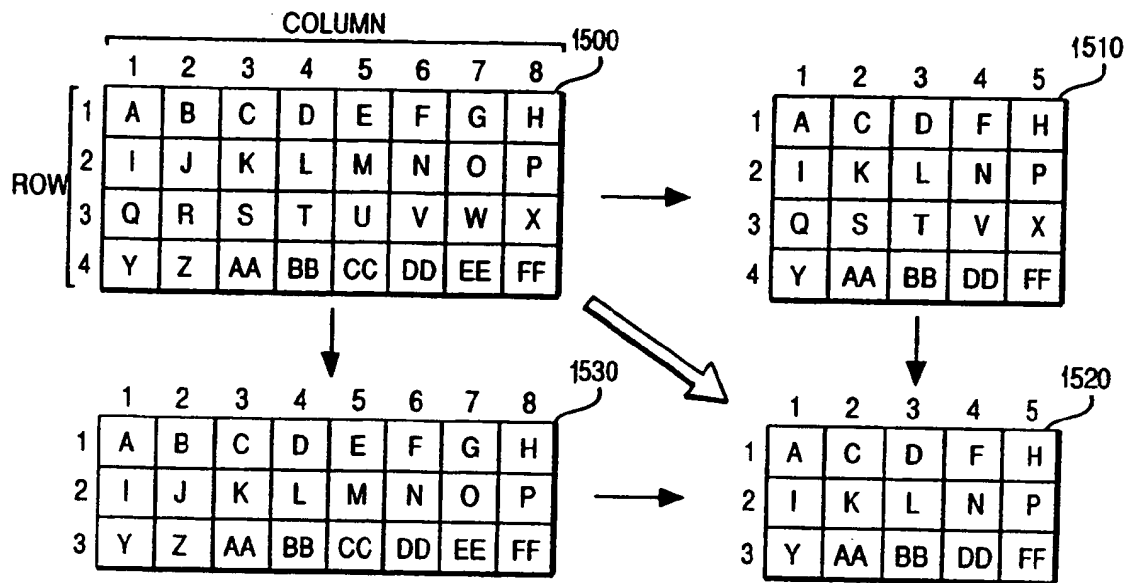


FIG. 9

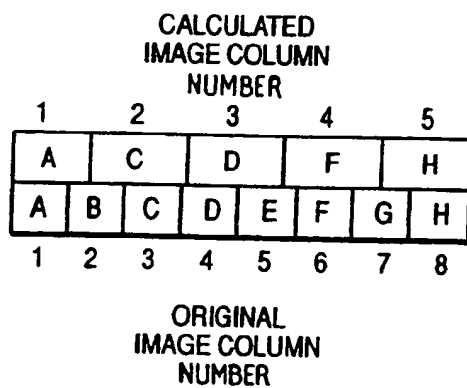


FIG. 10

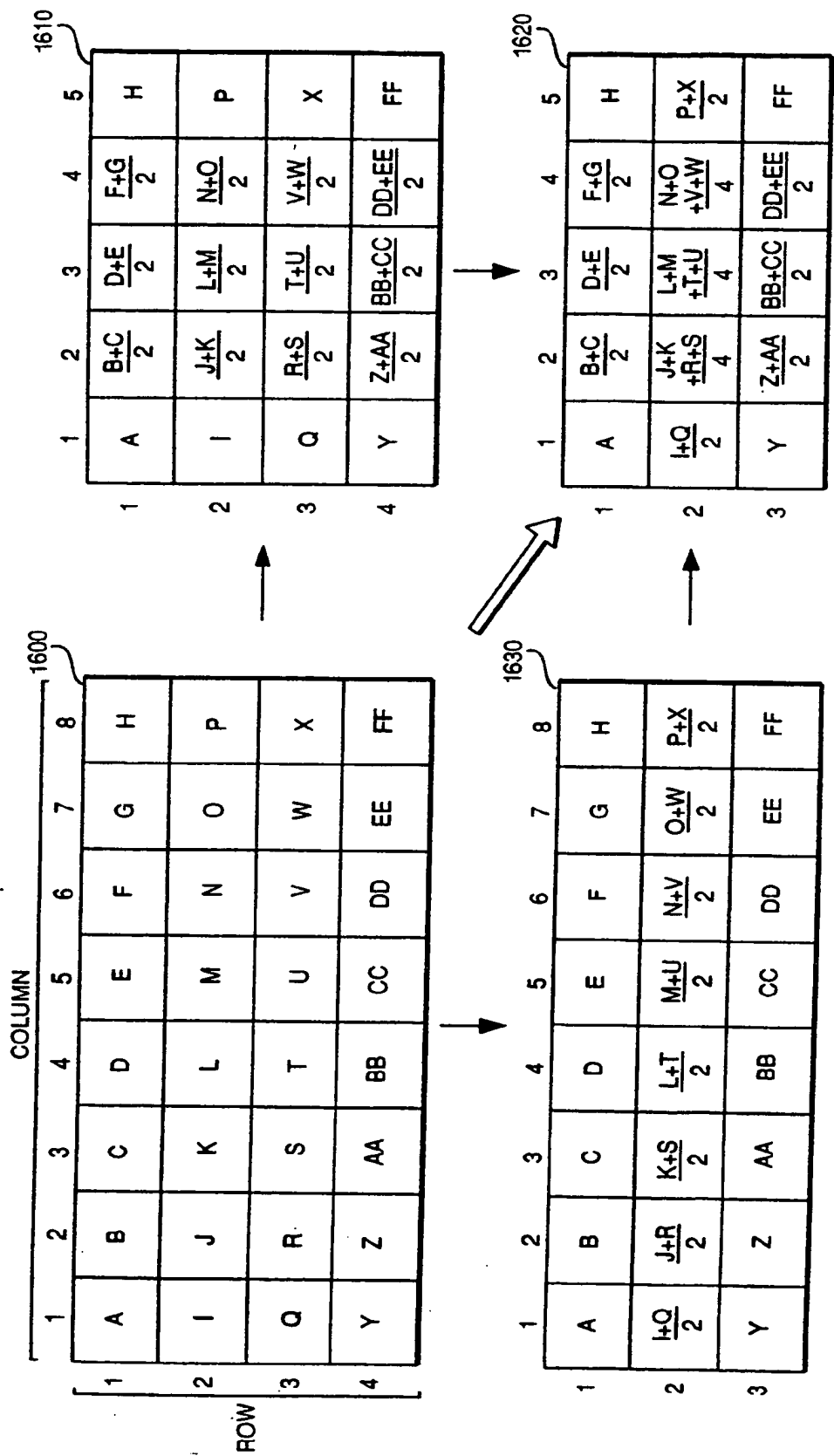


FIG. 11

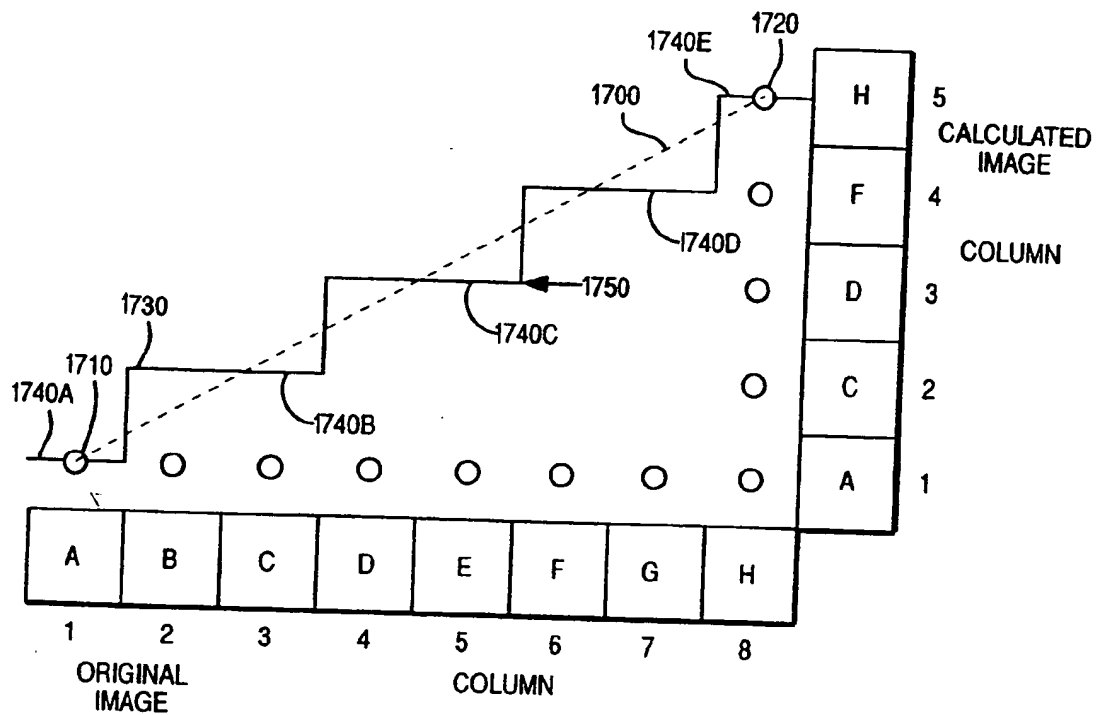


FIG. 12

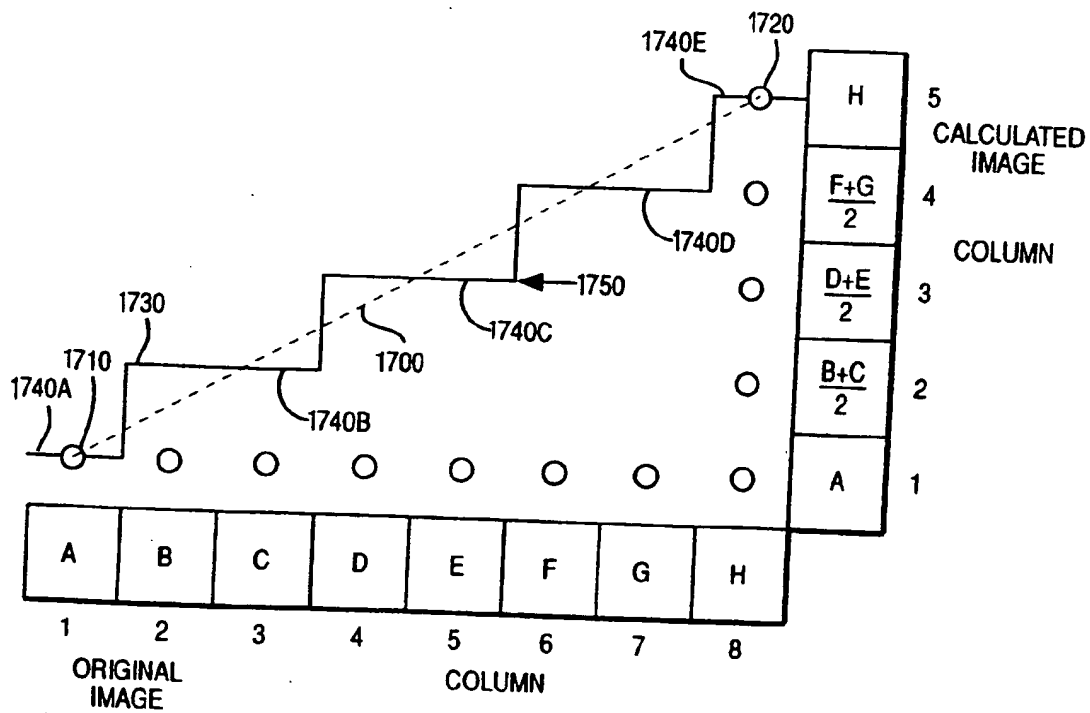


FIG. 13

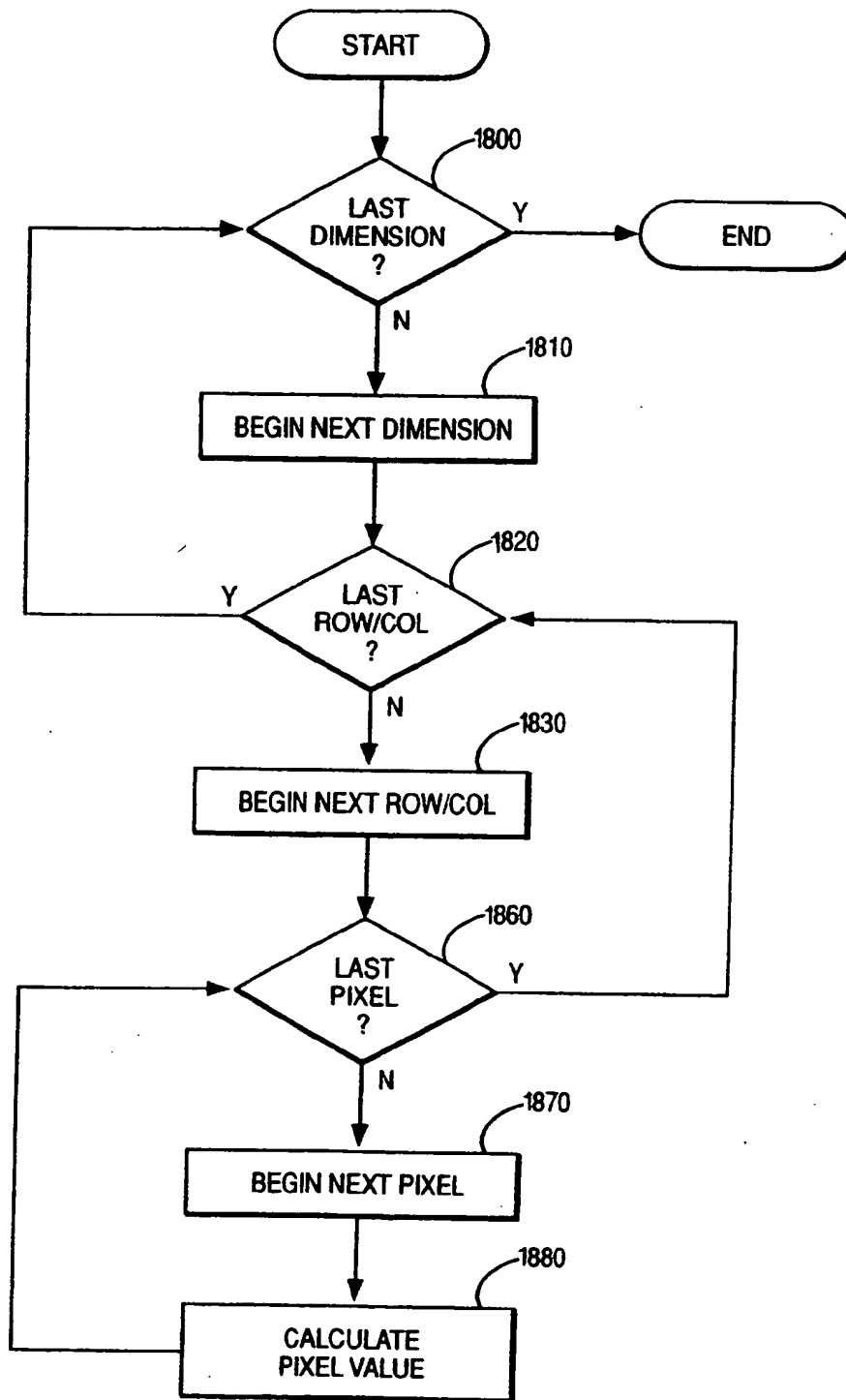


FIG. 14



(11) Publication number : **0 574 245 A3**

(12) **EUROPEAN PATENT APPLICATION**

(21) Application number : **93304476.0**

(51) Int. Cl.<sup>5</sup> : **G06F 15/62**

(22) Date of filing : **09.06.93**

(30) Priority : **11.06.92 US 897180**  
**11.06.92 US 897181**

(43) Date of publication of application :  
**15.12.93 Bulletin 93/50**

(84) Designated Contracting States :  
**DE FR GB**

(88) Date of deferred publication of search report :  
**18.05.94 Bulletin 94/20**

(71) Applicant : **International Business Machines Corporation**  
**Old Orchard Road**  
**Armonk, N.Y. 10504 (US)**

(72) Inventor : **Moller, Christian Henrik Luja**  
**11612 Buttonwood Drive**  
**Austin, Texas 78759 (US)**

(74) Representative : **Moss, Robert Douglas**  
**IBM United Kingdom Limited Intellectual**  
**Property Department Hursley Park**  
**Winchester Hampshire SO21 2JN (GB)**

(54) **Method and apparatus for variable expansion and variable shrinkage of an image.**

(57) Disclosed are methods and apparatus for variable magnification and for variable shrinking of an image. The techniques allow rapid zooms or rapid shrinks of images to be achieved by either hardware or software. The individual elements of the source and destination image data are correlated and then the remaining elements of the source data in the case of a shrinkage and of the destination data in the case of an expansion are distributed among the destination data or the source data respectively.

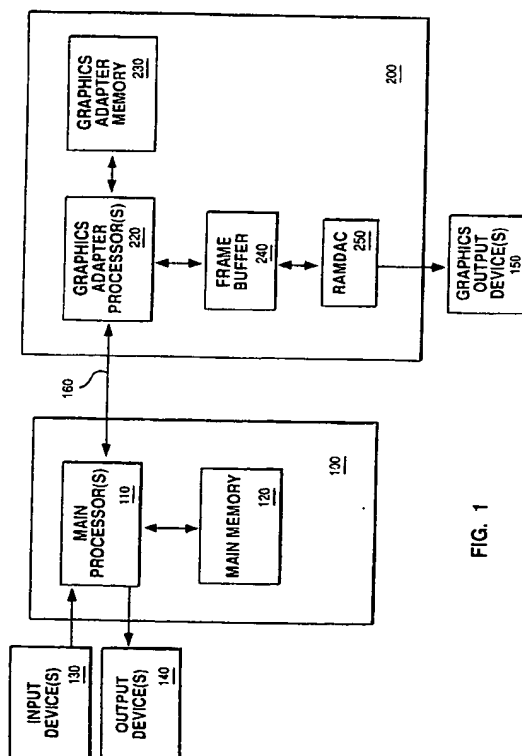


FIG. 1

EP 0 574 245 A3



European Patent  
Office

# EUROPEAN SEARCH REPORT

Application Number  
EP 93 30 4476

DOCUMENTS CONSIDERED TO BE RELEVANT			
Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int. Cl. 5)
X	PROCEEDINGS OF TENCON 87: 1987 IEEE REGION 10 CONFERENCE 'COMPUTERS AND COMMUNICATIONS TECHNOLOGY TOWARD 2000' (CAT. NO.87CH2423-2), SEOUL, SOUTH KOREA, 25-28 AUG. 1987, 110 - 114 VOL.1 Hasan S M R 'A novel LSI architecture for image scaling' * page 110, left column, line 1 - line 10 * page 111, right column, line 38 - line 60 *	1-16	G06F15/62
X	WO-A-90 16035 (EASTMAN KODAK CO ) 27 December 1990 * page 3, line 10 - line 33; figure 1 *	1-16	
A	US-A-4 661 987 (ANDERSON VERNON A ET AL) 28 April 1987 * column 1, line 65 - column 2, line 9 *	1-16	
A	FR-A-2 565 004 (ELSCINT LTD ) 29 November 1985 * page 3, line 27 - page 4, line 29; figure 2 *	1-16	TECHNICAL FIELDS SEARCHED (Int. Cl. 5) G06F
A	PATENT ABSTRACTS OF JAPAN vol. 012, no. 396 (P-774) 21 October 1988 & JP-A-63 136 275 (KONICA CORP) 8 June 1988 * abstract *		
The present search report has been drawn up for all claims			
Place of search THE HAGUE		Date of completion of the search 21 March 1994	Examiner Pierfederici, A
<p><b>CATEGORY OF CITED DOCUMENTS</b></p> <p>X : particularly relevant if taken alone Y : particularly relevant if combined with another document of the same category A : technological background O : non-written disclosure P : intermediate document</p> <p>T : theory or principle underlying the invention E : earlier patent document, but published on, or after the filing date D : document cited in the application L : document cited for other reasons A : member of the same patent family, corresponding document</p>			

EPO FORM 1500 (03.82) (P4/C01)